Tech Bulletin

Volume 6, Number 1 Spring 1998

Adding a Swap File to the Index Server

by Melissa Langston and Lois Bostain, PRC Inc.

For optimum system performance a filesystem should have swap space equal to two times the main memory. For example, if the disk has 256 MB of main memory, there should be 512 MB of swap space. In addition, the filesystem would normally be partitioned so that the swap space resides in one partition. If your system needs more swap space and re-partitioning the filesystem is not possible or convenient at this time, the best strategy is to create a swap file.

Create a swap file on an internal disk rather than on the RAID system. If the internal disk on your Index Server is less than 4 GB, it may not be possible to have two times the memory for swap. In that case, add a swap file to make the total swap space on the system as close as possible to twice the memory, but at least equal to it. Use the *hinv* command to check the amount of memory on your system.

# **hinv | grep memory**
# **hinv -c disk** to list the disk drives on your system

The internal drive on your Index Server might be **unit 1** on SCSI **controller 1** or unit 1 on SCSI controller 3. Controllers 2, 5, or 7 are used for external drives. In most cases, the RAID drives should be disk drive unit 2 on SCSI controller 7. There will also be several SCSI controllers listed that have no attached devices/disks. Note the controller number and unit number specified. Use that information to verify the existing amount of swap space and the drive partitioning information with the *fx* command. Remember that you are looking at a <u>live disk</u>, that is, all filesystems are still mounted and users are accessing the system as you verify this information and create the additional swap file. <u>Caution must be taken not to destroy these filesystems</u>. After you type *fx*, accept the system default for "device-name" and "lun#". Enter the appropriate value for `ctrl#` and `drive#`. In the following example, “1” was entered.

```
#  fx
   fx version 5.3, Sep 28, 1996
   fx: "device-name" = (dksc)
   fx: ctlr# = (0)  1
   fx: drive# = (1)  1
   fx: lun# = (0)
           opening dksc(1,1,0)

           fx: Warning:  this disk appears to have mounted filesystems.

           controller test...OK
           Scsi drive type == SGI     IBM  DFHSS2E    4C43

           —— please choose one (? for help, .. to quit this menu)——
           [exi]t                 [d]ebug/               [l]abel/
           [b]adblock/            [exe]rcise/            [r]epartition/
```

Select [r]epartition by typing **r** to see the existing partitions. Look for the raw partition as this is the type used for swap space. It is usually partition 1. Look at the Megabytes column.

continued on page 2

New Scripts to Identify Orphans

Two new scripts are being delivered with the JEDMICS 2.5.3 software release to assist in identifying orphan records whenever you use the index export utility. These scripts are located in the */edmics/sql* directory.

missing_dwg_research_to_accdoc.sql. Reports any mismatches between the records listed in the table ACC_DOCS and the table DWG_RESEARCH. Any orphan records in ACC_DOCS should be removed.

missing_dwg_research_to_mrc.sql. Reports any mismatches between the records listed in the table MAJOR_REPOSITORY_CODE and the table DWG_RESEARCH. Any orphan records in MAJOR_REPOSITORY_CODE should be removed. ♦

continued from page 1

The second number is the size. Most JEDMICS Index Servers have a 256 MB raw partition with a base of 100 rather than 26. Exit *fx* by typing **/exit**.

```
  fx> r
    —— partitions ——
   part  type       cyls          blocks       Megabytes  (base+size)
    0     efs        3  + 46      3183  + 49152        2  + 24
    1:    raw       51  + 494    54111  + 524288      26  + 256
    6:    efs      545  + 3605  578399  + 3825812    282  + 1868
    7:    efs        3  + 4148    3183  + 4401028      2  + 2149
    8:    volhdr     0  + 3          0  + 3183         0  + 2
    10:   volume     0  + 4151       0  + 4404211      0  + 2150

   capacity is 4404489 blocks

   —— please choose one (? for help, .. to quit this menu)——-
   [ro]otdrive        [u]srrootdrive     [o]ptiondrive      [re]size
   fx/repartition> /exit
```

Check the free space on your internal drive by using the *df -k* command.

```
#   df -k
```

The new swap file will be added to the */usr* partition. In the example, we are looking for 256 MB of free space. If you don't have enough space, see if you can move some user account home directories or other files to the external drives. You want enough freed space so that you will still have operating space after you make the new swap file. There are two options for the *swap* command. The **-***l* option produces output in blocks; the **-***ln* option in megabytes.

```
# swap -l
  lswap    path     dev    pri    swaplo   blocks   free    maxswap    vswap
    1   /dev/swap 128,273   0        0    524288   524288   524288      0
```

Blocks are 512 bytes (½ KB), which is the same as dividing by 2 to determine the number of kilobytes. To convert kilobytes to KB, divide by 1.024.

524288 / 2 = 262144 / 1.024 = 256,000KB = 256MB
installed as your primary swap space (lswap 1)

```
# swap -ln
       path    pri   pswap      free     maxswap   vswap
  1   dev/swap  0   256.00m   256.00m    256.00m   0.00k
```

Once you have freed up enough space to put in the swap file, use the *mkfile* command to create the file:

```
# mkfile 256m /usr/swap2
```

Next, manually add the swap file to the system using the **-***a* option and verify it has been added by using the **-***ln* option.

```
# swap -a /usr/swap2
# swap -ln
       path          pri       pswap    free      maxswap     vswap
  1   /dev/swap      0        256.00m   256.00m   256.00m     0.00k
  2   /usr/swap2     2        256.00m   256.00m   256.00m     0.00k
```

For the system to automatically mount the swap space at startup, use the *vi* editor to add a line to the end of */etc/fstab*. Then reboot the system.

continued on page 3

continued from page 2

```
  vi /etc/fstab
    /usr/swap2 swap swap pri=2 0 0

# reboot
```

Use the *swap* command to verify that the swap file was mounted at startup.

```
#   swap -ln
       path          pri       pswap    free      maxswap     vswap
  1   /dev/swap      0         256.00m   256.00m   256.00m     0.00k
  2   /usr/swap2     2         256.00m   256.00m   256.00m     0.00k
```
♦

Database Resizing

Recently a number of sites have had to resize their Oracle IMS database. At some point in time, each site will need to perform this task. An increase in the number of users, devices, or images will eventually cause one or more tables to outgrow the original limit that was set for each, or some segments of the database will become fragmented and system performance will slowly deteriorate. Importing a large number of images from another site will have the same effect if the database hasn't been sized to accommodate them. The purpose of this article is to help system administrators evaluate the health and integrity of their database and understand the various tasks that should be performed in preparation for resizing a database.

Identify and Remove Orphan Records

To check the integrity of your database, search for orphan records by running the "missing..." scripts listed below and located in */edmics/sql*. You will need to manually remove any orphan records that are identified.

- *missing_acc_docs_to_index.sql*
- *missing_docset_detail_to_index.sql*
- *missing_dwg_research_to_accdoc.sql*
- *missing_dwg_research_to_index.sql*
- *missing_dwg_research_to_mrc.sql*
- *missing_dwg_research_to_naval.sql*
- *missing_index_to_dwg_research.sql*
- *missing_naval_to_dwg_research.sql*
- *missing_rev_order_to_index.sql*

Clean Out the Dynamic Tables

Various tables increase in size as images are scanned and stored in the database. These tables grow slowly and are called static tables. Dynamic tables increase in size very quickly. They can easily exceed the space limitations set for them if they are not cleaned out regularly. These tables are:

- function_transactions
- session_transactions
- security_audit
- queue_master
- queue_detail
- migration_control
- ds_batch_log

In many cases, the data in these tables must be maintained for two or three months, primarily for use in monthly and quarterly status reports. If these tables are very fragmented and the data in them must be kept for a period of time, resizing them is probably a good idea. The best time to determine the size required for these tables would be just before you clean them out so they will be sized for the amount of data that meets your reporting needs.

How's Your Swap Space?

Adequate swap space is necessary to maintain proper operating system performance and can have an effect on how your SGI and Oracle perform. Ideally it should be twice the size of main memory. In some cases, particularly if your boot drive is only 2 GB, this amount may not be feasible. Then, swap space should be at least the same size as memory. Typically, to increase swap space a system administrator would perform an export of the database and a full system backup. Then the filesystem would be unmounted and repartitioned. Finally, the database would be imported. This process can be time consuming, particularly if a problem occurs. You may want to consider creating a swap file instead. See the article on Adding a Swap File on the front page of this issue.

continued on page 4

l

p

continued from page 3

Increase the Shared Pool Size

The shared pool size in the Oracle System Global Area (SGA) determines how much of the system's memory Oracle is allowed to use/access. If your database has grown or if system performance appears to be somewhat degraded, this value should be increased to equal 20% of the total memory available. For example, for a small or medium installation with 32-64 users and system memory of 256 MB, this value can be increased to 50 MB (entered as 50000000 with no commas). Log in as user oracle and type the following:

```
dbs1$  cp initIMS.ora  initIMS_orig.ora
dbs1$  vi /oracle/dbs/initIMS.ora
```

Make the changes on the "shared pool size" line that is not commented out and then save the file.

Checking Database Fragmentation

Fragmentation is a sign that the database is beginning to outgrow the size limitations set when it was originally created. Serious fragmentation can cause system performance to be degraded and productivity to suffer. Below are two SQL scripts you can run on a monthly basis to check the health of your database. Although they can be stored and run from any directory, it is suggested they be stored outside of the */edmics* and */oracle* directories. The files they create are written to */tmp*. Log into SQL*Plus and run them from the SQL prompt by entering "@<script_name>". Before you run these scripts, be sure the environment variables in the file */oracle/make/env.csh* are correct, and then add the following line to the end of the *.cshrc* file in */usr/edmics/jedadm*:

source /oracle/make/env.csh

jedtabs.sql script

```
rem  This script generates a list of all tables and all
       indexes along with
rem  their respective storage parameters. It generates two
       files,
rem  /tmp/tables.out and /tmp/indexes.out.

set  echo off
set  feedback off
set  pagesize 24
set  pause off

col tablespace_name format a17 heading Tablespace
col table_name format a20 heading Table_Name
col initial_extent format a10 heading Initial(k)
col next_extent format a10 heading Next(K)
col pct_increase format a2 heading PI
col count(T.table_name)format 999 heading Exts
col totalsize format a10 heading Size(K)

break on tablespace_name skip 1

spool /tmp/tables.out
SELECT   T.tablespace_name,
         T.table_name,
         COUNT(T.table_name),
         to_char(T.initial_extent/1024) as initial_extent,
         to_char(T.next_extent/1024) as next_extent,
         to_char(T.pct_increase) as pct_increase,
         to_char(SUM(E.bytes)/1024) as totalsize
FROM sys.dba_extents E, dba_tables T
WHERE E.segment_name = T.table_name
AND T.owner = upper('EDMICS')
AND E.segment_type = 'TABLE'
GROUP BY T.table_name,T.initial_extent,
          T.next_extent,T.pct_increase,T.tablespace_name
ORDER BY 1,3 desc,2;

spool off
col index_name format a20 heading Table_Name
col count(I.index_name) format 999 heading Exts

spool/tmp/indexes.out

SELECT   I.tablespace_name,
         I.index_name,
         COUNT(I.index_name),
         to_char(I.initial_extent/1024)as initial_extent,
         to_char(I.next_extent/1024) as next_extent,
         to_char(I.pct_increase) as pct_increase,
         to_char(SUM(E.bytes)/1024) as totalsize
FROM sys.dba_extents E, dba_indexes I
WHERE E.segment_name = I.index_name
AND I.owner = upper('EDMICS')
AND E.segment_type = 'INDEX'
GROUP BY I.index_name,I.initial_extent,I.next
_extent,I.pct_increase,I.tablespace_name
ORDER BY 1,3 desc,2;

spool off

clear column
clear breaks
```

mapper.sql script

```
rem This script prompts for a tablespace name and then
       generates a 'map'
rem of that tablespace, showing objects and free space.
       This map is
rem written into a file "/tmp/<tablespace_name>.lst" - for
       example,
rem /tmp/drawing.lst.

prompt Enter tablespace_name:
set termout off
select '&&tablespace_name' from dual;
set termout on

set pause off
set pagesize 20 linesize 80 verify off
column file_id heading "File|Id"

spool /tmp/&tablespace_name

SELECT
         'free space'owner, /*owner of free space */
         ' 'object,         /*blank object name*/
         file_id,           /* file ID for extent header*/
         block_id,          /*block ID for extent header */
         blocks             /*length of the extent,
                              (blocks)*/

FROM sys.dba_free_space
WHERE tablespace_name=upper('&&tablespace_name')
UNION
SELECT
         substr(owner,1,15),   /*owner name(lst 20 charac-
         ters)*/
         substr(segment_name,1,25),   /*segment name*/
         file_id,
         block_id,
         blocks
FROM sys.dba_extents
WHERE tablespace_name=upper('&&tablespace_name')
ORDER BY 3,4

spool off
undefine tablespace_name
set pause off
```

If there are two or more extents on an index, you may be able to just drop the index and recreate it to eliminate the extents. If there are two or more extents on a table, you might need to resize that table as well as the tablespace it is in and then adjust the index as needed. That process will be covered in a future issue of the Tech Bulletin. ♦

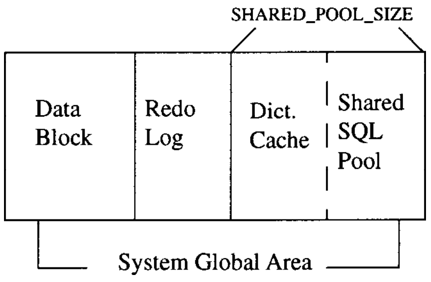
Tuning the System Global Area (SGA)

by Joe Stroy, PRC Inc.

As a system administrator one of your ongoing tasks is to monitor and optimize the performance of the database. One part of that optimization is the sizing of the System Global Area (SGA). This article provides some general information about the SGA, a procedure for changing parameters, and finally, a tool to monitor the results of your changes.

The SGA is a segment of memory allocated to Oracle that contains data and control information particular to an Oracle instance (IMS). The SGA facilitiates the transfer of information between users. It also holds the most commonly requested structural information about the database. SGA is divided into four main sections in Oracle7.

- Data Block Buffers
- Redo Log Buffers
- Dictionary Cache
- Shared SQL Pool



The *Tech Bulletin* is published by the JEDMICS Program Management Office, PRC Inc., 12005 Sunrise Valley Drive, Reston, VA 20191-3423.

Managing Editor, Kathleen C. Altizer, JEDMICS Training and Site Support Manager

Editor, Anne Shepard, Support Products

If you have questions or comments about the *Tech Bulletin*, or would like to contribute information or articles, contact Anne Shepard at (703) 264-5521 or at shepard_annette@prc.com.

Data Block Buffers

The data block buffers are a cache in the SGA used to hold the data blocks that are read from the data segments in the database, such as tables, indexes, and clusters. The size of the data block buffer cache is determined by the DB_BLOCK_BUFFERS parameter in the *initIMS.ora* file for the Index Server. Managing the size of the data block buffer cache plays an important part in managing and tuning the database.

Redo Log Buffers

Redo log files hold entries that describe the changes that are made to the database. These changes are written to the online redo log files. They can be used in roll-forward operations during database recoveries before being written to the online redo log files. However, they are first cached in the SGA in an area called the redo log buffer. The database periodically writes batches of redo entries to the online redo log files, thus optimizing this operation. The size (in bytes) of the redo log buffers is set by the LOG_BUFFER parameter in the *initIMS.ora* parameter file.

Dictionary Cache

Information about database objects is stored in data dictionary tables. This information includes user account data, datafile names, segment names, extent locations, table descriptions, and privileges. As needed by the database, these data dictionary tables are read and the data that is returned is stored in the SGA in the dictionary cache. The size of the dictionary cache is managed internally by the database; it is part of the shared SQL pool whose size is set by the SHARED_POOL_SIZE parameter in the *initIMS.ora* file.

Shared SQL Pool

In Oracle7 databases, the dictionary cache is stored in an area called the shared SQL pool. This memory area also includes information about statements that are run against the database. Thus, while the data block buffer and dictionary cache enable sharing of structural and data information between users in the database, the shared SQL pool allows the sharing of commonly used SQL statements.

The shared SQL pool contains the execution plan and parse tree for SQL statements run against the database. The second time that an identical SQL statement is run (by any user) it is able to take advantage of the parse

continued on page 6

continued from page 5
information available in the shared SQL pool to expedite its execution.

Questions to Ask before Changing SGA

Now that you know what the SGA does, the question now is how much memory should be allocated to the SGA in order to achieve the desired performance goals? How do you know if the SGA is performing at its optimum? What tools do you use to tune or monitor the performance of the SGA?

Sizing the SGA

Normally the size of the SGA used to stored shared memory objects for the database is set once, and rarely monitored or tuned after that. However, the proper sizing of the SGA is critical to the performance of the database. Sizing of the SGA is partially a hit-and-miss exercise. You will find that some parameters in the initialization parameter file have a profound effect on the size of the SGA. You will see that DB_BLOCK_BUFFERS and SHARED_POOL_SIZE are the major contributors to the size of the SGA.

You may allocate up to 80% of system RAM to the SGA. However, you must consider the overall performance of the server, memory overhead per user, network of operating system buffers and other system needs that could be impacted if there is not enough available memory. This means that sizing the SGA is hardly an exact science. Most of the JEDMICS systems were configured for a SHARED_POOL_SIZE=9000000. This is 9 MB. The total system RAM in some cases is equal to 512 MB. To adjust the SHARED_POOL_SIZE, edit the */oracle/dbs/initIMS.ora* file and set the SHARED_POOL_SIZE parameter to the desired value. Anytime a change is made to the *initIMS.ora* file shut down the Oracle database and restart it. Then check the SHARED_POOL_SIZE in Oracle by logging onto the database as “sqldba”. Then type the command **show sga**. A sample output is as follows:

```
SQLDBA> show sga
Total System Global Area    5887388 bytes
                Fixed Size    52500 bytes
             Variable Size  4544648 bytes
          Database Buffers  1126400 bytes
              Redo Buffers   163840 bytes
```

Other individual initialization parameters that are impacted by the SGA size are DB_BLOCK_BUFFERS and LOG_BUFFER. Oracle automatically adjusts the dictionary cache based on the memory allocated to the SHARED_POOL_SIZE. If you have 512 MB of system RAM you can set the SHARED_POOL_SIZE=102000000. If your system RAM is equal to 384 MB set the SHARED_POOL_SIZE=77000000. This is a starting point that is approximately 20% of the system RAM.

Performance Monitoring

It is important to know how the SGA is performing before and after altering the above settings. Without this information, it would be difficult to know how the database performance is affected.

Oracle continuously updates a set of internal statistics. These statistics should only be stored in tracking tables when you can be sure that the database will not be shut down between monitoring checks. This is necessary because these internal statistics are reset each time the database is shut down and restarted. In order to facilitate monitoring of these statistics, Oracle provides two scripts named UTLBSTAT.SQL and UTLESTAT.SQL. These scripts are located in the */oracle/rdbms/admin* directory. UTLBSTAT (Begin Statistics) creates a set of tables and populates them with the statistics in the database at that time it is executed. The second file, UTLESTAT.SQL run at a later time, creates a set of tables based on the statistics in the database at that time it is run. Then the script generates a report (REPORT.TXT) that lists the changes in the statistics during the interval between the run times for the beginning and the ending scripts. Before starting this process, be sure TIME_STATISTICS=TRUE in the *initIMS.ora* file.

The report that is generated must then be analyzed to determine if the results are meeting the performance goals as specified by Oracle. This process must then be repeated until the desired system results are achieved.

Additional Resources

Oracle DBA Handbook by Oracle Press
Oracle8 Tuning by Oracle Press
Tuning Oracle by Oracle Press ♦

JRTS News Release

by Mark Madalena, PRC Inc.
Look for changes to the JEDMICS home page in the following months. The home page is being redesigned for easier navigation. The home page has SPR Web Query capabilities, Tech Bulletins, TECPs, ECPs, specifications, and other valuable information. The JEDMICS home page is located at http://jrts.jedmics.navy.mil—let us know what you think. Send e-mail to the Webmaster with any JRTS suggestions or ideas at feedback@jrts.jedmics.navy.mil.
A new JRTS Schema was made available on the JEDMICS home page in April. If you have any problems with the installation of the new schema, contact the JRTS Help Desk at (703) 620-8226 or send e-mail to feedback@jrts.jedmics.navy.mil.
Connectivity issues can be submitted online from the JEDMICS home page. ♦